Enhanced algorithms for Local Search

Yves F. Verhoeven*

Abstract

Let $G = (V, E)$ be a finite graph, and $f : V \to \mathbb{N}$ be any function. The Local Search problem consists in finding a *local minimum of the function f on G*, that is a vertex v such that $f(v)$ is not larger than the value of f on the neighbors of v in G. In this note, we first prove a separation theorem slightly stronger than the one of Gilbert, Hutchinson and Tarjan for graphs of constant genus. This result allows us to enhance a previously known deterministic algorithm for Local Search with query complexity $O(\log n) \cdot d + O(\sqrt{g}) \cdot \sqrt{n}$, so that we obtain a deterministic query complexity of $d + O(\sqrt{g}) \cdot \sqrt{n}$, where n is the size of G, d is its maximum degree, and g is its genus. We also give a quantum version of our algorithm, whose query complexity is of $O(\sqrt{d}) + O(\sqrt[4]{g}) \cdot \sqrt[4]{n} \log\log n$. Our deterministic and quantum algorithms have query complexities respectively smaller than the algorithm Randomized Steepest Descent of Aldous and Quantum Steepest Descent of Aaronson for large classes of graphs, including graphs of bounded genus and planar graphs. Independently from this work, Zhang has recently given a quantum algorithm which finds a local minimum on the planar grid over $\{1, \ldots, \sqrt{n}\}^2$ using $O(\sqrt[4]{n}(\log\log n)^2)$ queries. Our quantum algorithm can be viewed as a strongly generalized, and slightly enhanced version of this algorithm.

1 Introduction

The Local Search problem consists in finding a *local minimum of the function f on G*, that is a vertex v such that $f(v)$ is not larger than the value of f on the neighbors of v in G. Obviously, such a vertex always exists, as a global minimum satisfies this constraint. Another easy argument shows how to find such a vertex : make a walk over vertices such that at each step the next vertex is the neighbor of the current vertex which has the smallest value; the walk will stop in a local minimum. Such a walk is called a *steepest descent*. Steepest descents are the basis of several approaches to efficiently find a local minimum.

The Local Search problem has been previously studied and there is already a large literature on its complexity. Its structural complexity, where the function and the graph are given as an input to a Turing machine, was studied in [7, 10], and its query complexity, where the graph is known but the values of f are accessed through an oracle, was investigated in [9, 8, 2, 1, 11, 12]. We focus on the query complexity, which complexity is obviously at most the size of the graph. Our query model is the standard one; see Section 3.2 for precise definitions.

The deterministic query complexity of Local Search on a graph G of size n, maximum degree d and genus g (for a definition of the genus, see for instance [4]), is intimately connected to the size of *separators* of G:

Definition 1. A *separator for G* is a subset of V whose removal leaves no connected component with more than $2n/3$ vertices.

In [9], a deterministic query algorithm was exhibited, which works using a sub-linear number of

*Laboratoire de Recherche en Informatique, Bâtiment 490, Université Paris-Sud, 91405 Orsay Cedex, France and École Nationale Supérieure des Télécommunications, 46 rue Barrault, 75013 Paris, France. E-mail: yves.verhoeven@normalesup.org. The research was supported by the EU 5th framework program RESQ IST-2001-37559, and by the ACI CR 2002-40 and ACI SI 2003-24 grants of the French Research Ministry.

queries for large classes of graphs: a local minimum can be found on the graph G using $O(\log n) \cdot d + O(g) \cdot \sqrt{n}$ queries. It is based on the recursive use of separators for smaller and smaller subgraphs of G, and their complexity analysis relies on the following result:

Theorem 1 (Gilbert, Hutchinson, Tarjan [6]). *The graph G has a separator of size at most $6\sqrt{gn} + 2\sqrt{2n} + 1$.*

In the randomized and quantum query models, the situation is quite different, as the size of separators of G. Also, the only known sub-linear query algorithm for general graphs is a randomized algorithm, that we call Randomized Steepest Descent, was exhibited by Aldous [2], and has a query complexity $\Theta(\sqrt{nd})$. The idea of this algorithm is to choose $\sqrt{nd}$ vertices at random, query their values, start a steepest descent from the vertex with smallest value for at most $\sqrt{nd}$ steps, and to return the last visited vertex. This idea was later refined by Aaronson [1] to give a sub-linear quantum query algorithm for general graphs, that we call Quantum Steepest Descent, using $\Theta(n^{1/3}d^{1/6})$ queries.

On the side of lower-bounds, it follows from [9] that the size of a smallest separator is a lower-bound on the deterministic complexity of Local Search. Also, from [1], we know that d is a lower-bound on the deterministic query complexity, $\Omega(d)$ a lower-bound for the randomized query complexity, and $\Omega(\sqrt{d})$ a lower-bound for the quantum query complexity.

2 Results

In this note, we first improve Theorem 1 in Section 4.1, to obtain the following slightly stronger separation theorem:

Theorem 2 (strong separation for graphs of genus g). *Assume $n \geq 3$. There exists a separator C for G such that C contains no more than $(6 + 2\sqrt{2} - 12/n + 6\sqrt{g} + 4g/n + 1/\sqrt{n}) \cdot \sqrt{n}$ vertices, and the subgraph induced on G by $V \setminus C$ has maximal degree at most $\sqrt{n}$.*

As a result it allows us to enhance, in Section 5, the deterministic algorithm of Llewellyn, Tovey and Trick [9] whose complexity is of $O(\log n) \cdot d + O(\sqrt{g}) \cdot \sqrt{n}$. We also derive a quantum algorithm from it. More precisely, we obtain the following result:

Theorem 3. *There exists a deterministic and a quantum query algorithms that find a local minimum of f on G using respectively $d + O(\sqrt{g}) \cdot \sqrt{n}$ and $O(\sqrt{d}) + O(\sqrt[4]{g}) \cdot \sqrt[4]{n} \log\log n$ queries.*

Our deterministic and quantum algorithms have smaller query complexities than the respective algorithms Randomized Steepest Descent of Aldous [2] and of Quantum Steepest Descent of Aaronson [1] for large classes of graphs, including graphs of bounded genus and planar graphs. We analyze this in detail in Section 6.

Independently from this work, Zhang [12] has recently given an algorithm which finds a local minimum on the planar grid over $\{1, \ldots, \sqrt{n}\}^2$ using $O(\sqrt[4]{n}(\log\log n)^2)$ queries. Our quantum algorithm can be viewed as a strongly generalized, and slightly enhanced version of this algorithm.

3 Preliminaries

3.1 Notations

We denote by $\log n$ the natural logarithm of n, and for every positive real number b we denote by $\log_b n$ the logarithm of n in base b. If G is a graph and v is any vertex of G, we denote by $\partial_G(v)$ the set of neighbors of v in G.

3.2 Query complexity

In the query model of computation we count only queries made by the algorithm, but all other computations are free. The state of the computation is represented by three registers, the query register $i \in \{1, \ldots, n\}$, the answer register $a \in \Sigma$, and the work register $z \in W$, where Σ and W are finite sets. The computation takes place in the vector space spanned by all basis states $|i\rangle|a\rangle|z\rangle$. In the *quantum*

query model introduced by Beals, Buhrman, Cleve, Mosca and de Wolf [3] the state of the computation is a complex combination of all basis states which has unit length for the norm ℓ_2, and the allowed operations on the state of the computation are all isometric operators for the ℓ_2 norm acting over the computation space. In the randomized model, the state of the computation is a non-negative real combination of all basis states of unit length for the norm ℓ_1, and the allowed operations on the state of the computation are all isometric operators for the norm ℓ_1 acting over the computation space. In the deterministic model, the state of the computation is always one of the basis states, and the allowed operations are all operators mapping a basis state to another basis state.

Assume that $x \in \Sigma^n$ is the input of the problem which can be accessed only through the oracle. The query operation $\mathcal{O}_x$ is the permutation which maps the basis state $|i\rangle|a\rangle|z\rangle$ into the state $|i\rangle|(a+x_i) \bmod |\Sigma|\rangle|z\rangle$ (here we identify Σ with the residue classes $\bmod|\Sigma|$). Non-query operations are independent of x. A *k-query algorithm* is a sequence of $(k+1)$ operations $(U_0, U_1, \ldots, U_k)$ where U_i is an allowed operation in the chosen model of computation. Initially the state of the computation is set to some fixed value $|0\rangle|0\rangle|0\rangle$, and then the sequence of operations $U_0, \mathcal{O}_x, U_1, \mathcal{O}_x, \ldots, U_{k-1}, \mathcal{O}_x, U_k$ is applied. The final state is denoted by Φ.

The output in the quantum model is an element $z \in W$ that appears with probability equal to the square of the ℓ_2 norm of the orthogonal projection of Φ over the vector space V spanned by $\{|i\rangle|a\rangle|z\rangle \mid i \in \{1, \ldots, n\},\ a \in \Sigma\}$. The output in the randomized model is an element $z \in W$ that appears with probability equal to the ℓ_1 norm of the orthogonal projection of Φ over the vector space V. The output in the deterministic model is the element $z \in W$ such that there exist i and a with $\Phi = |i\rangle|a\rangle|z\rangle$.

Assume that $R \subseteq \Sigma^n \times W$ is a total relation (*i.e.* for every $x \in \Sigma^n$ there exists $z \in W$ such that $(x, z) \in R$) that we want to compute. A quantum or randomized algorithm computes (with two-sided error) R if its output yield some $z \in W$ such that $(x, z) \in R$ with probability at least $2/3$. A deterministic algorithm computes R if its output yield some $z \in W$ such that $(x, z) \in R$.

Then the query complexity of a relation R in a model of computation (deterministic, randomized or quantum) is the smallest k for which there exists a k-query algorithm, in that model of computation, which computes R.

4 Tools

In this section, we recall and prove the results that we need in order to design the algorithms of Section 5.

4.1 Separation in graphs of higher genus

We first recall the following well-known theorem for graphs of higher genus (see for instance [4]) :

Theorem 4. *Any n-vertex graph of genus g with $n \geq 3$ contains no more than $3n - 6 + 2g$ edges.*

This result, together with Theorem 1, allows us to prove Theorem 2.

Proof of Theorem 2. Theorem 1 shows that it is possible to find a separator C' for G such that C' has size at most $6\sqrt{gn} + 2\sqrt{2n} + 1$. Let B be the set of all vertices of degree greater than $\sqrt{n}$. Using Theorem 4, we have

$$|B| \cdot \sqrt{n} \leq \sum_{v \in V} \mathrm{d}(v) = 2 \cdot |E| \leq 6n - 12 + 4g.$$

The set $C = C' \cup B$ being a superset of a separator of G is also a separator of G. From the definition of B, the subgraph induced on G by $V \setminus C$ obviously has maximal degree at most $\sqrt{n}$. □

The genus of a graph being in $O(n^2)$, the asymptotic inequality $g/n = O(\sqrt{g})$ holds and therefore Theorem 2 can be interpreted as stating the existence of a particular $O(\sqrt{g}) \cdot \sqrt{n}$ separator G.

4.2 Minimum-finding algorithms

In this paragraph, we recall results about the query complexity of finding the minimum value of a function on a set.

Let n be a positive integer, S be a set of cardinality n, $g : S \to \mathbb{N}$ be a function and $\mathcal{O}_g$ be an oracle for g.

Definition 2. Let $\varepsilon < 1$ be any positive real number. If $\mathcal{A}$ is a randomized algorithm that outputs the minimum value of the function g on S with probability at least $1 - \varepsilon > 0$, then we denote by $\operatorname{argmin}_{\mathcal{A}}\{g(s) \mid s \in S\}$ the random variable equal to its output.

It is obvious that, for every deterministic algorithm $\mathcal{A}$, computing $\operatorname{argmin}_{\mathcal{A}}\{g(s) \mid s \in S\}$ requires querying all n values of g to $\mathcal{O}_g$. It is natural that, for every randomized algorithm $\mathcal{A}$, computing $\operatorname{argmin}_{\mathcal{A}}\{g(s) \mid s \in S\}$ requires querying $\Omega(n)$ values of g to $\mathcal{O}_g$. It is more surprising that much less queries are needed when quantum queries are allowed:

Theorem 5 (Dürr, Høyer [5]). *There exists a quantum algorithm which finds the minimum value of g with probability at least $1/2$, using $O(\sqrt{n})$ quantum queries to $\mathcal{O}_g$.*

Amplification of the probability of success of the algorithm of Dürr and Høyer can be obtained by running the algorithm several times, and then taking the minimum value of all the values that have been returned by each repetition of the algorithm. After k repetitions, the probability of having found the minimum is at least $1 - 2^{-k}$. In particular, for every positive real number $\varepsilon < 1$, there exists a quantum algorithm $\mathcal{A}$ computing $\operatorname{argmin}_{\mathcal{A}}\{g(s) \mid s \in S\}$ with probability at least $1-\varepsilon$ using $O(\sqrt{n}\log(1/\varepsilon))$ quantum queries.

5 Enhanced algorithms for Local Search

In this paragraph, we prove Theorem 3. The proof will be in two steps: in Theorem 6 we will prove the correction of our algorithms, and in Theorem 7 we will prove their complexity.

The basic procedure of our algorithms follows the lines of the algorithms of Llewellyn, Tovey and Trick of [9] and Santha and Szegedy [11]. It is given in Algorithm 1. The main idea is to adopt a divide-and-conquer approach: the graph is split into connected components of small size by removing a separator; then, querying the values of the vertices in and close to that separator make it possible to find one of these connected components in which there is a local minimum of f on G.

Notice that neither the way the separators are chosen, or how the minimum-finding algorithms $\mathcal{A}_i$ work for integers $i \geq 1$, are specified in Algorithm 1. Our algorithms consist in using the procedure described in Algorithm 1 with the following specific choices:

- a separator C of a graph G' will be chosen according to Theorem 2 if G' has more than two vertices, and C contains all vertices of G' otherwise,

- the minimum-finding algorithm $\mathcal{A}_i$ will behave as follows when requested to minimize the function f over a set $S \subseteq V$ of vertices: in the deterministic case, the local minimum of f is always found by exhaustive search[1]. In the quantum cases, if the set S has size at most 3, then the minimum value of f on S is also found using exhaustive search. Otherwise, the output is the one found by the last measurement at the end of the quantum procedure described in paragraph 4.2; moreover, we request that the minimum-finding algorithm $\mathcal{A}_1$ has error probability $1/12$ using $O(\sqrt{|S|})$ queries, and $\mathcal{A}_i$ has error probability $1/12\log_{3/2} n$ using $O(\sqrt{|S|}\log\log n)$ queries, for $i \neq 1$.

[1] One should observe that it is important at this point that f takes distinct values on distinct vertices. This can be assumed, as one could for instance put a total order $\prec$ on V, and minimize the function $g : v \mapsto (f(v), v)$ according to the lexicographic order induced on $\mathbb{N} \times V$ by $<$ and $\prec$, instead of minimizing f. The function g takes distinct values on distinct vertices.

Algorithm 1 Procedure for finding a local minimum of a function $F : V \to \mathbb{N}$ on a graph $G = (V, E)$, using separators.

$i := 0$, $G^{(0)} := G$, $v^{(0)} :=$ any vertex of G, output $:= \emptyset$.
while output $= \emptyset$ **do**
  $i := i + 1$.
  Create a separator $C^{(i)}$ for $G^{(i-1)}$.
  $m^{(i)} := \operatorname{argmin}_{\mathcal{A}_i}\{f(v) \mid v \in C^{(i)}\}$.
  $z^{(i)} := \operatorname{argmin}_{\mathcal{A}_i}\{f(v) \mid v \in \partial_{G^{(i-1)}}(m^{(i)})\}$.
  $v^{(i)} := \operatorname{argmin}_{\mathcal{A}_i}\{f(v) \mid v \in \{v^{(i-1)}, m^{(i)}, z^{(i)}\}\}$.
  if $v^{(i)} = m^{(i)}$ **then**
    output $:= \{v^{(i)}\}$.
  else
    $V^{(i)} :=$ the connected component of $V^{(i-1)} \setminus C^{(i)}$ that contains $v^{(i)}$.
    $G^{(i)} := G[V^{(i)}]$.
  end if
end while
Return output.

Theorem 6. *With our choice of minimum-finding algorithm, Algorithm 1 always returns a local minimum in the deterministic case, and returns a local minimum with probability at least $2/3$ in the quantum case.*

Proof. Let j be the largest value of the variable i for a run of the algorithm. First, an easy inductions shows that for every iteration $i \leq j$ of the main loop, and every $v \in V^{(i)}$ we have

$$\partial_G(v) \subseteq \partial_{G^{(i)}}(v) \cup C^{(1)} \cup C^{(2)} \cup \cdots \cup C^{(i)}.$$

So, to prove that f is minimized on $v^{(j)}$, one must only prove that $f(v^{(j)})$ is not larger than $\min\{f(v) \mid v \in \partial_{G^{(j)}}(v) \cup C^{(1)} \cup C^{(2)} \cup \cdots \cup C^{(j)}\}$.

If, during the run of the algorithm, the calls to the algorithms $\mathcal{A}_i$, for $1 \leq i \leq j$, have always successfully returned elements minimizing f, then for every positive integer $i \leq j$ we have $f(v^{(i)}) \leq \min\{f(v^{(i-1)}), f(m^{(i)}), f(z^{(i)})\}$. Therefore, an easy induction shows that $f(v^{(i)}) \leq \min\{f(v) \mid v \in C^{(k)}\}$, for every positive integers $k \leq i \leq j$. Moreover, the equality $v^{(j)} = m^{(j)}$ implies $f(v^{(j)}) \leq \min\{f(v) \mid v \in \partial_{G^{(j)}}(v^{(j)})\}$. So, if $\mathcal{A}_i$ never failed to find a minimizing element, then the criterion given in the previous paragraph shows that $v^{(j)}$ is a local minimum.

In the deterministic case, the algorithms $\mathcal{A}_i$, for $1 \leq i \leq j$, always return an element minimizing f, and therefore Algorithm 1 always returns a local minimum.

In the quantum case, a call to $\mathcal{A}_i$ returns an element minimizing f with error probability at most $1/12$ for $i = 1$, and at most $1 - 1/4\log_{3/2} n$ for $1 < i \leq j$. The set $C^{(i)}$ being a separator of $G^{(i-1)}$ for every positive integer $i \leq j$, we have $|V^{(i)}| \leq 2|V^{(i-1)}|/3$. This implies that $j \leq \log_{3/2} n$, and the probability that $\mathcal{A}_i$ did not return an element minimizing f at some point is at most $2 \cdot 1/12 + 2 \cdot \log_{3/2} n/(12\log_{3/2} n) = 1/3$. So, in the quantum case, Algorithm 1 returns a local minimum with probability at least $2/3$. □

Theorem 7. *With our choices of separators, Algorithm 1 has a deterministic query complexity at most $d + O(\sqrt{g}) \cdot \sqrt{n}$, and a quantum query complexity at most $O(\sqrt{d}) + O(\sqrt[4]{g}) \cdot \sqrt[4]{n} \log\log n$.*

Proof. Again, let j be the largest value of the variable i for a run of the algorithm. Let us denote by $\mathcal{C}_{\mathcal{A}_i}(s)$ the number of queries made by the minimum-finding algorithm $\mathcal{A}_i$ on a set of size s, and by $L^i(n, d)$ the number of queries that are made in the

i-th iteration of the main loop of our algorithm on a graph G' that has n vertices and is of maximum degree d. We denote also by d_i the maximum degree of $|G^{(i)}|$, for a non-negative integer $i \leq j$. Analysis of the main loop of Algorithm 1 gives, for every positive integer $i \leq j$,

$$L^i(n,d) \leq \mathcal{C}_{\mathcal{A}_i}(|C^{(i)}|) + \mathcal{C}_{\mathcal{A}_i}(d_{i-1}) + 3.$$

Let us denote by $T^i_\gamma(\alpha,\beta)$ the number of queries made by our algorithm in the main loop between its i-th iteration and the end of the algorithm if $i < j$ and 0 if $i \geq j$, on an input graph which has α vertices, is of maximum degree β and has genus at most γ. Theorem 2 ensures that for every positive integer $i \leq j$ we have $|V^{(i)}| \leq 2|V^{(i-1)}|/3$, and $d_i \leq \sqrt{|V^{(i-1)}|}$. Moreover, the genus of be $G^{(i)}$ is not larger than the genus of $G^{(i-1)}$. So, by induction we have $|V^{(i)}| \leq (2/3)^i n$, and the genus of $|G^{(i)}|$ is at most g. Therefore, for every integer $1 \leq i \leq j$ we have $|C^{(i)}| \leq O(\sqrt{g}) \cdot \sqrt{(2/3)^{i-1}n}$, $d_0 = d$ and $d_i \leq \sqrt{(2/3)^{i-1}n}$. This leads to the following equations:

$$\begin{aligned} &T^1_g(n,d) \\ &\quad \leq L^1(n,d) + T^2_g(n,d) \\ &\quad \leq \mathcal{C}_{\mathcal{A}_1}(O(\sqrt{g}) \cdot \sqrt{n}) + \mathcal{C}_{\mathcal{A}_1}(d) + 3 + T^2_g(n,d), \end{aligned}$$

and for every $i \in \{2, \ldots, \lfloor \log_{3/2} n \rfloor - 1\}$,

$$\begin{aligned} &T^i_g(n,d) \\ &\quad \leq L^i(n,d) + T^{i+1}_g(n,d) \\ &\quad \leq \mathcal{C}_{\mathcal{A}_i}\left(O(\sqrt{g}) \cdot \sqrt{(2/3)^{i-1}n}\right) + \\ &\qquad \mathcal{C}_{\mathcal{A}_i}\left(\sqrt{(2/3)^{i-2}n}\right) + 3 + T^{i+1}_g(n,d). \end{aligned}$$

In the deterministic case we have $\mathcal{C}_{\mathcal{A}_i}(k) = k$ for all positive integers k and i, and in the quantum case we have, for all positive integer k, $\mathcal{C}_{\mathcal{A}_i}(k) = O(\sqrt{k})$ when $i = 1$, and $\mathcal{C}_{\mathcal{A}_i}(k) = O(\sqrt{k}\log\log n)$ when $i \neq 1$. So, in the deterministic case, summing all the previous inequalities gives

$$\begin{aligned} T^1_g(n,d) \leq O(\sqrt{g}) \cdot \sqrt{n} \cdot \sum_{i=0}^{\infty} \sqrt{2/3}^i + d + \\ \sqrt{n} \cdot \sum_{i=0}^{\infty} \sqrt{2/3}^i + 3\log_{3/2} n + \\ T_g^{\lfloor \log_{3/2} n \rfloor}(n,d), \end{aligned}$$

which shows $T^1_g(n,d) = d + O(\sqrt{g}) \cdot \sqrt{n}$, as $T_g^{\lfloor \log_{3/2} n \rfloor}(n,d) = O(1)$, and the query complexity of our deterministic algorithm is $T^1_g(n,d) = d + O(\sqrt{g}) \cdot \sqrt{n}$. In the quantum case, it gives

$$\begin{aligned} T^1_g(n,d) \leq O(\sqrt[4]{g}) \cdot \sqrt[4]{n}\log\log n \cdot \sum_{i=0}^{\infty} \sqrt[4]{2/3}^i + \\ O(\sqrt{d}) + O(\sqrt[4]{n}\log\log n) \cdot \sum_{i=0}^{\infty} \sqrt[4]{2/3}^i + \\ 3\log_{3/2} n + T_g^{\lfloor \log_{3/2} n \rfloor}(n,d), \end{aligned}$$

leading to a quantum query complexity $T^1_g(n,d) = O(\sqrt{d}) + O(\sqrt[4]{g}) \cdot \sqrt[4]{n}\log\log n$. □

6 Comparison with generic algorithms

Let us first compare the query complexity of our deterministic algorithm with the query complexity of the algorithm Randomized Steepest Descent of Aldous [2]. The complexity of our algorithm is $d + O(\sqrt{g}) \cdot \sqrt{n}$, and the complexity of Randomized Steepest Descent is $\Theta(\sqrt{nd})$. As $d \leq n$, our algorithm performs as well as Randomized Steepest Descent (up to a constant speedup factor) as soon as $g = O(d)$, and performs asymptotically better when $g = o(d)$. In particular, our deterministic algorithm has lower query complexity than Randomized Steepest Descent on classes of graphs with bounded genus, which includes the class of planar graphs.

Let us now compare the query complexity of our quantum algorithm with the query complexity

of the algorithm Quantum Steepest Descent of Aaronson [1]. The complexity of our algorithm is $O(\sqrt{d}) + O(\sqrt[4]{g}) \cdot \sqrt[4]{n} \log\log n$, and the complexity of Quantum Steepest Descent is $\Theta(n^{1/3}d^{1/6})$. As $d \leq n$, we have $\sqrt{d} \leq n^{1/3}d^{1/6}$, and our algorithm performs as well as Quantum Steepest Descent (up to a constant speedup factor) as soon as $g^{1/2} \cdot n^{1/4} \log\log n = O(n^{1/3}d^{1/6})$, that is to say $g = O(n^{1/6}d^{1/3}/(\log\log n)^2)$. This holds if $g = O(\sqrt{d}/(\log\log n)^2)$. Also, our quantum algorithm performs asymptotically better when $g = o(n^{1/6}d^{1/3}/(\log\log n)^2)$, and therefore when $g = o(\sqrt{d}/(\log\log n)^2)$. In particular, our quantum algorithm has lower query complexity than Quantum Steepest Descent on classes of graphs with bounded genus, which includes the class of planar graphs.

In conclusion, the algorithms we have designed perform better than the known generic algorithms for some classes of graphs, in particular planar graphs and graphs of constant genus, both for classical (deterministic and randomized) computation, and for quantum computation.

References

[1] Scott Aaronson. Lower bounds for local search by quantum arguments. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 465–474. ACM Press, 2004.

[2] David Aldous. Minimization algorithms and random walk on the d-cube. *Ann. Probab.*, 11(2):403–413, 1983.

[3] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001.

[4] Béla Bollobás. *Modern graph theory*, volume 184 of *Graduate Texts in Mathematics*. Springer-Verlag, 1998.

[5] Christoph Dürr and Peter Høyer. A quantum algorithm for finding the minimum. quant-ph/9607014.

[6] John R. Gilbert, Joan P. Hutchinson, and Robert Endre Tarjan. A separator theorem for graphs of bounded genus. *J. Algorithms*, 5(3):391–407, 1984.

[7] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *J. Comput. System Sci.*, 37(1):79–100, 1988.

[8] Donna C. Llewellyn and Craig A. Tovey. Dividing and conquering the square. *Discrete Appl. Math.*, 43(2):131–153, 1993.

[9] Donna C. Llewellyn, Craig A. Tovey, and Michael Trick. Local optimization on graphs. *Discrete Appl. Math.*, 23(2):157–178, 1989.

[10] Christos H. Papadimitriou, Alejandro A. Schaëffer, and Mihalis Yannakakis. On the complexity of local search. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 438–445. ACM Press, 1990.

[11] Miklos Santha and Mario Szegedy. Quantum and classical query complexities of local search are polynomially related. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 494–501. ACM Press, 2004.

[12] Shengyu Zhang. (Almost) tight bounds for randomized and quantum local search on hypercubes and grids. quant-ph/0504085.